Math 29: The Recursion Theorem

April 15th, 2022

1 Recursion Theorem

The Recursion Theorem allows us to create programs which are self-referential. One can think of this as similar to recursion in the computer science sense: for example, we can program the `factorial` function to, on input n, call itself with input with new input $n-1$ and multiply the return value by n. As long as we provide a base case to stop the recursion from proceeding infinitely, this will halt and compute $n!$. The Recursion Theorem allows us to do even more powerful things, like performing operations on the index of the function we are defining.

In computability, given a function $f : \omega \to \omega$, e is a **fixed point** of f if $\varphi_e = \varphi_{f(e)}$.

Theorem 1. *(The Recursion Theorem) Every total computable function $f : \omega \to \omega$ has a fixed point. Moreover, given an index for f, we can uniformly compute its fixed point.*

Proof: Consider the partial computable function g such that $g(x, y) = \varphi_{f(\varphi_x(x))}(y)$. By the s-m-n theorem, there is a total computable function s such that

$$\varphi_{f(\varphi_x(x))}(y) = \varphi_{s(x)}(y)$$

Then let m be an index for s, which we can compute using an index for f.

Now notice that $\varphi_m(m)\downarrow$, as $\varphi_m(m) = s(m)$, and s is total. Thus

$$\varphi_{s(m)}(y) = \varphi_{\varphi_m(m)}(y) = \varphi_{f(\varphi_m(m))}(y)$$

Then $\varphi_m(m)$ is a fixed point of f.

The recursion theorem originates from an attempt to diagonalize out of the class of computable functions. This was originally done by Kleene, who was attempting to disprove the Church-Turing thesis. That this fails provides evidence of the contrary: that we should believe the Church-Turing thesis.

Specifically, we look at the following matrix:

$$\begin{array}{ccccc} \varphi_0(0) & \varphi_0(1) & \varphi_0(2) & \varphi_0(3) & \dots \\ \varphi_1(0) & \varphi_1(1) & \varphi_1(2) & \varphi_1(3) & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \varphi_x(0) & \varphi_x(1) & \varphi_x(2) & \varphi_x(3) & \dots \\ \dots & \dots & \dots & \dots & \dots \end{array}$$

To try and diagonalize out of this list, we try and look at the function f which uses the universal machine to compute $f(n)$ using the machine $\varphi_n(n)$, i.e. using the machine whose code appears on the n-th row in the above table in the diagonal slot. (Or $f(n)$ diverges if $\varphi_n(n)$ does.) f is intuitively computable, so if it is not in the above list, then the Church-Turing thesis fails. The idea is that, if e is an index for f, then

$$f(e) = \varphi_e(e) = \varphi_{\varphi_e(e)}(e)$$

On the surface, this seems like it should not be true.

However, f is in this list, and we can compute exactly which row it is using the method in the proof of the recursion theorem. Even more generally, if we try to computably pick and choose rows to diagonalize out of this list, then the function we create will still be in this list, and we will still be able to compute the fixed point in the same way. This is very strong evidence that one cannot diagonalize out of the partial computable functions.

The recursion theorem allows us to show the existence of some surprising computable functions.

Lemma 2. *There is an e such that $\varphi_e(n) = n + e$.*

Proof: By the s-m-n theorem, there is a total computable function s such that $\varphi_{s(x)}(y) = x + y$. By the recursion theorem, s has a fixed point e. Then

$$\varphi_e(y) = \varphi_{s(e)}(y) = y + e$$

Lemma 3. *There is an e such that $W_e = \{e\}$.*

Proof: Recall that W_x is the domain of φ_x. Therefore, we need to find an e such that $\varphi_e(e) \downarrow = 1$ and $\varphi_e(k) \uparrow$ for $k \neq e$. Consider the partial computable function $f(x, y)$ such that $f(x, y) = 0$ if $x = y$ and $f(x, y) \uparrow$ otherwise. Let m be an index for f.

By the s-m-n theorem, there is a total computable function s such that $\varphi_{s(m,x)}(y) = \varphi_m(x, y)$. Let $g : \omega \to \omega$ be defined via $g(k) = s(m, k)$. Then g is a single-variable, total computable function, so it has a fixed point e. That is,

$$\varphi_e(y) = \varphi_{g(e)}(y) = \varphi_{s(m,e)}(y) = \varphi_m(e, y)$$

for all y. Then, in particular, $\varphi_e(e) = \varphi_m(e, e) \downarrow = 1$, but for $k \neq e$, $\varphi_e(k) = \varphi_m(e, k) \uparrow$.

Thus $W_e = \{e\}$ as desired.

The trick we used in the proof above shows us how to improve the recursion theorem to allow for parameters.

Theorem 4. *(Recursion Theorem with Parameters) If $f(y, x_0, \ldots, x_{k-1})$ is a total computable function, then there is an injective, total computable function $r(x_0, \ldots, x_{k-1})$ such that*

$$\varphi_{r(x_0,\ldots,x_{k-1})} = \varphi_{f(r(x_0,\ldots,x_{k-1}),x_0,\ldots,x_{k-1})}$$

Proof: By the s-m-n theorem, there is an injective, total computable function $d : \omega^{k+1} \to \omega$ such that

$$\varphi_{d(y,x_0,\ldots,x_{k-1})}(n) = \begin{cases} \varphi_{\varphi_y(y,x_0,\ldots,x_{k-1})}(n) & \text{if} \varphi_y(y, x_0, \ldots, x_{k-1}) \downarrow \\ \uparrow & \text{otherwise} \end{cases}$$

Let e be an index of $f(d(y, x_0, \ldots, x_{k-1}))$, and define the total computable function $r : \omega^k \to \omega$ via $r(x_0, \ldots, x_{k-1}) = d(e, x_0, \ldots, x_{k-1})$. Then we claim that $r(x_0, \ldots, x_{k-1})$ is a fixed point.

Observe that

$$\varphi_{r(x_0,\ldots,x_{k-1})}(n) = \varphi_{d(e,x_0,\ldots,x_{k-1})}(n)$$

By the definition of d and choice of e, this is equal to

$$\varphi_{\varphi_e(e,x_0,\ldots,x_{k-1})}(n) = \varphi_{f(d(e,x_0,\ldots,x_{k-1}),x_0,\ldots,x_{k-1})}(n)$$

Thus this proves the claim, and we are done. (Notice that we need not worry about the second case in the original definition of d, as e is an index of the total function $f \circ d$.)

Essentially, the recursion theorem allows us to define a computable function **using its index in the definition.** While on the surface this is circular reasoning, behind the scenes we can define a function with an extra parameter using the s-m-n theorem, then apply the recursion theorem to show that there is an index which is equal to plugging itself in as the added parameter. This can all be done computably. Furthermore, the recursion theorem with parameters even allows us to do so with free variables floating around and still be able to compute the index uniformly in those variables.

In fact, there will be infinitely many indices we can use to satisfy the recursion theorem, so we are not able to try and change the function slightly to break the fixed point.

Lemma 5. *Each total computable function has infinitely many fixed points.*

Proof. Homework 3 Question 6. □